

Huistaak - Macro's

Oefening 1

Schrijf een macro `NIEUW` dat een nieuw element (`struct verpl`) alloceert op de `HEAP` en dit gepast initialiseert.

```
| struct verpl {  
|   int van;  
|   int naar;  
|   int aantal;  
|   struct verpl * volg;  
| };  
| De macro nieuw alloceert een record van het type struct verpl op de HEAP  
| en initialiseert de velden met de meegegeven parameters  
| De parameters van, naar en aantal zijn *geheugenadressen* van de velden waar  
|   deze waarden gevonden kunnen worden  
| De parameter Lijst is een *register* dat het adres van een lijst bevat  
| Het gealloceerde record wordt vooraan in de lijst geplaatst!  
| De parameter Reg geeft aan welk *register* mag gebruikt worden
```

MACRO

```
    NIEUW van, naar, aantal, Lijst, Reg
```

```
    ...
```

MCREINDE

Gebruik de macro in het vertaalde C-programma (torens van Hanoi van enkele weken geleden, zie ook op achterzijde) waarbij de oproep naar de functie `nieuw` vervangen wordt door een oproep van de macro `NIEUW`.

Welke versie van het programma zal sneller uitvoeren? Waarom? Vergelijk ook de lengte van de vertaalde programma's. Welk is langer? Verklaar!

De C-code voor de torens van Hanoi (zie oefenzitting enkele weken terug):

```
struct verpl {
    int van;
    int naar;
    int aantal;
    struct verpl * volg;
};
// De geschreven macro vervangt de volgende functie
struct verpl * nieuw (int van, int naar, int aantal,
                      register struct verplaatsing * lijst) {
    register struct verpl * v = (struct verpl *) alloc(sizeof(struct verpl));
    v->van = van;
    v->naar = naar;
    v->aantal = aantal;
    v->volg = lijst;    // plaats het element vooraan in de lijst
    return (v);
}
main () {
    int aantal;
    register struct verpl * lijst;
    register struct verpl * huidig;
    int via, aVerpl;
    aantal = getint();
    // verplaats AANTAL ringen van 1 naar 2
    // !!! *Hier wordt de macro NIEUW gebruikt* !!!
    lijst = nieuw(1, 2, aantal, NULL);
    aVerpl = 0;
    while (lijst != NULL) {
        huidig = lijst;
        lijst = lijst->volg; // haal element uit de lijst
        if (huidig->aantal == 1) {
            printint(huidig->van, huidig->naar);
            aVerpl++;
        } else {
            via = 6 - huidig->van - huidig->naar;
            /* genereer in omgekeerde volgorde en voeg vooraan toe aan lijst */
            // !!! *Hier zal telkens de macro gebruikt worden* !!!
            lijst = nieuw (via, huidig->naar, huidig->aantal - 1, lijst);
            lijst = nieuw (huidig->van, huidig->naar, 1, lijst);
            lijst = nieuw (huidig->van, via, huidig->aantal - 1, lijst);
        }
    }
    printint(aVerpl);
}
```